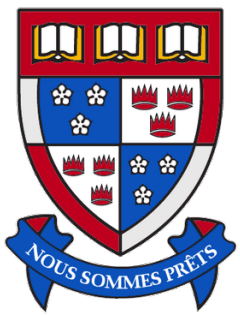
# A Modular Algorithm to Compute the Resultant of Multivariate Polynomials over Algebraic Number Fields Presented with Multiple Extensions

Mahsa Ansari
Michael Monagan

Department of Mathematics,
Simon Fraser University,
Canada



# Background

## Algebraic Number Field

Let $\alpha_1, \ldots, \alpha_n$ be algebraic numbers. The **algebraic number field** $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$ is the smallest field containing $\mathbb{Q}$ and $\alpha_1, \ldots, \alpha_n$.

## Theorem (1)

*Let $\alpha_1, \ldots, \alpha_n$ be algebraic numbers. There exists $\theta \in \mathbb{C}$ s.t*

$$\mathbb{Q}(\alpha_1, \ldots, \alpha_n) = \mathbb{Q}(\theta).$$

*We call $\theta$ a primitive element.*

## Example

$\mathbb{Q}(\sqrt{2}, \sqrt{3}) = \mathbb{Q}(\sqrt{2} + \sqrt{3})$.

# Resultant

Let $f_1; f_2 \in R[x_1; \ldots; x_k][y]$ and $f_1 = \sum_{j=0}^{m} a_j y^j$, $f_2 = \sum_{j=0}^{n} b_j y^j$.

$$\mathrm{sylv}(f_1; f_2; y) = \begin{bmatrix} a_m & & a_0 & \\ & \ddots & & \ddots \\ & a_m & & a_0 \\ b_n & & b_0 & \\ & \ddots & & \ddots \\ & & b_n & & b_0 \end{bmatrix}$$

## Resultant

The resultant of $f_1$ and $f_2$ w.r.t. the variable $y$ is defined as

$$res(f_1; f_2; y) = \det(\mathrm{sylv}(f_1; f_2; y)) \in R[x_1; \ldots; x_k].$$

# Why computing resultants is important?

Resultants appear as a subproblem in

- Factorization of polynomials over algebraic fields using Trager's algorithm.
- Solving systems of multivariate polynomials.
- Elimination theory.

# History

## 1971, Collins

A modular algorithm to compute the resultant of multivariate polynomials over $\mathbb{Z}$.

# History

## 1971, Collins

A modular algorithm to compute the resultant of multivariate polynomials over $\mathbb{Z}$.

## 2002-Monagan and Van Hoeij

A modular GCD algorithm for polynomials in $\mathbb{Q}(x_1, \ldots, x_n)[x]$.

# History

## 1971, Collins

A modular algorithm to compute the resultant of multivariate polynomials over $\mathbb{Z}$.

## 2002-Monagan and Van Hoeij

A modular GCD algorithm for polynomials in $\mathbb{Q}(\ _1, \ldots, \ _n)[x]$.

## 2023-Ansari and Monagan

A modular GCD algorithm that reduces the gcd problem over $\mathbb{Q}(\ _1, \ldots, \ _n)$ to gcd calculation over $\mathbb{Q}(\ )$ where $\ $ is a primitive element of $\mathbb{Q}(\ _1, \ldots, \ _n)$.

# Our Contributions

Let $f_1; f_2 \in \mathbb{Q}( \ _1; : : : ; \ _n)[x_1; : : : ; x_k; y]$.

$MRES(f_1; f_2; y)$ computes $r = \mathrm{res}(f_1; f_2; y)$.

Employing the monic Euclidean algorithm for $f_1; f_2 \in \mathbb{Z}_p[ \ _1; : : : ; \ _n][y]$, we present a new formula for computing $\mathrm{res}(f_1; f_2; y)$.

A Maple implementation by using RECDEN, a Maple library for recursive dense representation for polynomials.

The expected time complexity analysis.

A partial failure probability analysis.

# Computation over $\mathbb{Q}(\alpha_1, \ldots, \alpha_n)$

Let

$$\begin{aligned} L_0 &= \mathbb{Q} \\ L_i &= L_{i-1}\langle M_i(z_i)\rangle \quad \text{for } 1 \leq i \leq n \\ L_n &= L \end{aligned}$$

where $M_i(z_i)$ is the minimal polynomial of $\alpha_i$ over $L_{i-1}$.

$$\mathbb{Q}(\alpha_1, \ldots, \alpha_n) = L$$

**Remark**

MRES assumes that we are given the minimal polynomials $M_i(z_i)$.

In order to improve computational efficiency, in a preprocessing step in MRES, we eliminate fractions from the input polynomials $f_1$ and $f_2$ and the minimal polynomials $M_1, \ldots, M_n$.

## Example

Let $f = \frac{3}{2}\ _1x + \ _2 \in \mathbb{Q}(\ _1, \ _2)[x]$ where $\ _1 = \sqrt{2}$ and $\ _2 = \sqrt{3}$. Eliminate fractions from $f$, we have

$$f = 3\ _1x + 2\ _2.$$

# Monic Euclidean Algorithm (MEA)

**Algorithm 2:** Monic Euclidean Algorithm

**Input:** $f_1, f_2 \in R[x]$ such that $0 \leq \deg(f_2) \leq \deg(f_1)$ and R is a commutative ring with identity $1 \neq 0$.

**Output:** Either the monic $\gcd(f_1, f_2)$ or FAIL.

1 $r_1, r_2 = f_1, f_2$
2 $M_1, i = r_1, 2$
3 **while** $r_i \neq 0$ **do**
4 $\quad M_i = monic(r_i)$
5 $\quad$ **if** $M_i = failed$ **then return** *(FAIL)* `// The algorithm encountered a zero-divisor.`
6 $\quad$ Set $r_{i+1}$ to be the remainder of $M_{i-1}$ divided by $M_i$
7 $\quad$ Set $i = i + 1$
8 $l = i - 1$
9 **return**$(M_l)$

**Note:**

MEA fails if a remainder $r_i$ exists, such that $\mathsf{lc}(r_i)$ is not invertible over $R$. In other words, if $M_i = (r_i)$ fails at Step 4, then MEA returns FAIL.

## m.p.r.s.

Given $f_1; f_2\ 2\ R[x]$ with $\deg(f_2)\quad \deg(f_1)$, assume that the Monic Euclidean Algorithm (MEA) does not fail for $f_1$ and $f_2$ and terminates after $l$ iterations. We define the Monic Polynomial Remainder Sequence, m.p.r.s., generated by polynomials $f_1$ and $f_2$ as the sequence $r_1; r_2; \ldots; r_l; r_{l+1}$ obtained from the execution of the Monic Euclidean Algorithm such that $r_1 = f_1$, $r_2 = f_2$, $r_3 = r_1 \quad M_2 q_3$, and $r_{i+1} = M_{i\ 1} \quad M_i q_{i+1}$ with $M_i = monic(r_i)$ and $\deg(r_{i+1}) < \deg(r_i)$ for $2 \quad i \quad l \quad 1$ and $r_{l+1} = 0$.

# Using MEA to Compute Resultant

## Theorem

*Suppose that* $f_1; f_2 \in R[x]$ *and the Monic Euclidean Algorithm does not fail for* $f_1$ *and* $f_2$. *Let* $r_1; r_2; \ldots; r_l; r_{l+1}$ *be the m.p.r.s. generated by* $f_1$ *and* $f_2$ *where* $r_{l+1} = 0$. *Let* $n_i = \deg(r_i)$ *for* $1 \quad i \quad l$. *If* $\deg(r_l) > 0$, *then* $\operatorname{res}(f_1; f_2) = 0$. *Otherwise, we have*

$$\operatorname{res}(f_1; f_2) = (\quad 1)^v \Big(\prod_{i=2}^{l-1} \operatorname{lc}(r_i)^{n_i \quad 1}\Big) \operatorname{lc}(r_l)^{n_l \quad 1}$$

*where* $v = \sum_{i=1}^{l \quad 2} n_i n_{i+1}$.

We can modify the MEA to compute the resultant of two univariate polynomials $f_1; f_2 \; 2 \; R[x]$ in algorithm URES.

**Algorithm 3:** URES

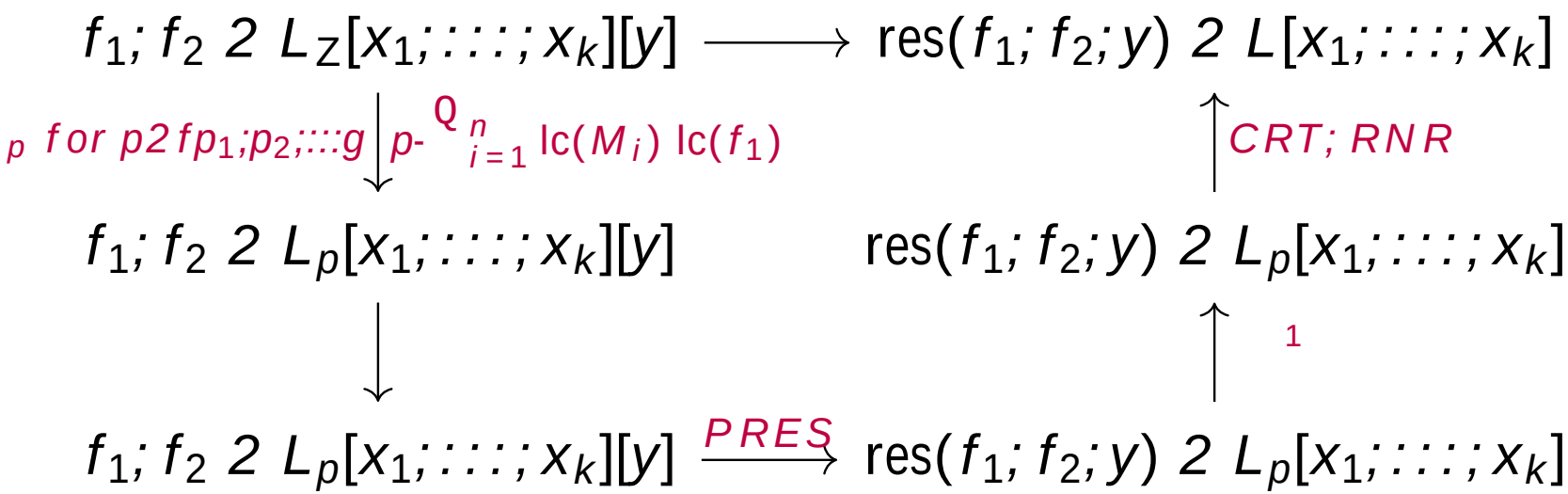
**Require:** $f_1, f_2 \in R[x]$ such that $0 \leq \deg(f_2) \leq \deg(f_1)$ where $R$ is a commutative ring with identity $1 \neq 0$.

**Ensure:** Either $\text{res}(f_1, f_2)$ or FAIL.

1: $r_1 = f_1$, $r_2 = f_2$, $i = 2$
2: $M_1 = r_1$, $R = 1$, $v = 0$
3: $n_1 = \deg(f_1)$ ,$n_2 = \deg(f_2)$
4: **while** $r_i \neq 0$ **do**
5: $M_i = monic(r_i)$
6: **if** $M_i = failed$ **return** *(FAIL)*// The algorithm encounters a zero-divisor.
7: Set $r_{i+1}$ to be the remainder of $M_{i-1}$ divided by $M_i$
8: Set $n_{i+1} = \deg(r_{i+1})$
9: **if** $n_{i+1} < 0$ and $n_i \neq 0$ **then return**(0) // If $\gcd(f_1, f_2)$ is not a constant, then $\text{res}(f_1, f_2) = 0$
10: Set $R = R \cdot \text{lc}(r_i)^{n_i - 1}$
11: Set $v = v + n_i n_{i-1}$
12: Set $i = i + 1$
13: **end while**
14: $R = (-1)^v R$
15: **return**(R)

# MRES

Let $f_1; f_2 \in L[x_1; \ldots; x_k][y]$, MRES computes $\mathrm{res}(f_1; f_2; y)$.

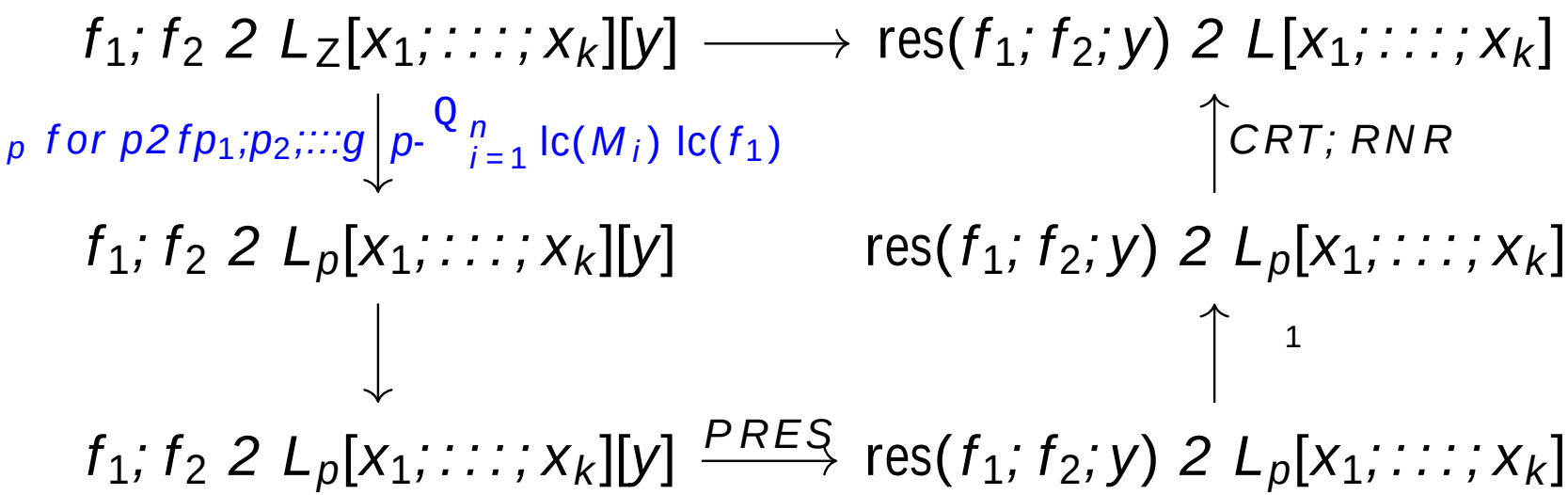



$L = \mathbb{Q}[z_1; \ldots; z_n]/\langle M_1(z_1); \ldots; M_n(z_n)\rangle$
$L_Z = \mathbb{Z}[\ _1; \ldots; \ _n]$
$L_p = \mathbb{Z}_p[z_1; \ldots; z_n]/\langle m_1(z_1); \ldots; m_n(z_n)\rangle$ s.t $m_i(z_i) = M_i(z_i) \mod p$
$L_p = \mathbb{Z}_p[z]/\langle M(z)\rangle$

# Reduction mod $p$

## The modular homomorphism

$_p : Z \quad ! \quad Z_p$ maps integers into their remainder modulo p. We choose p to be a prime so $Z_p$ is a finite field.




$L_p = Z_p[z_1;:::;z_n]=\langle m_1(z_1);:::;m_n(z_n)\rangle$ s.t $m_i(z_i) = M_i(z_i) \mod p$

# Lc-bad, Det-bad, and Zero-divisor

**Question:** Can we use any prime?

**Answer:** No!!!

# Lc-bad, Det-bad, and Zero-divisor

Let $f_1; f_2 \in L[x_1; \ldots; x_k][y]$ and $p$ be a prime.

Lc-bad Prime: If $p$ divides either $\mathsf{lc}(f_1)$, $\mathsf{lc}(f_2)$, or any $\mathsf{lc}(M_i(z_i))$ for $1 \leq i \leq n$, we call $p$ an lc-bad prime.

Det-bad Prime: Let $B_{Lp}$ be a basis of $L_p$ and $d$ be a primitive element and $A = \begin{bmatrix} \vdots & & \vdots \\ \cdots & \cdots & \cdots \\ \vdots & & \vdots \end{bmatrix}_{B_{Lp} \times B_{Lp}}$.

If $\det(A) \bmod p = 0$, then $p$ is called a det-bad prime.

Zero-divisor Prime: If $p$ is neither an lc-bad nor a det-bad prime and algorithm URES fails over $L_p$, then $p$ is called a zero-divisor prime.

Good Prime: If $p$ is neither lc-bad, det-bad nor a zero-divisor prime, we define it as a good prime.

## Example

Let $L = \mathbb{Q}[z_1; z_2]/\langle z_1^2 - 2; z_2^2 - 3i \rangle$ and

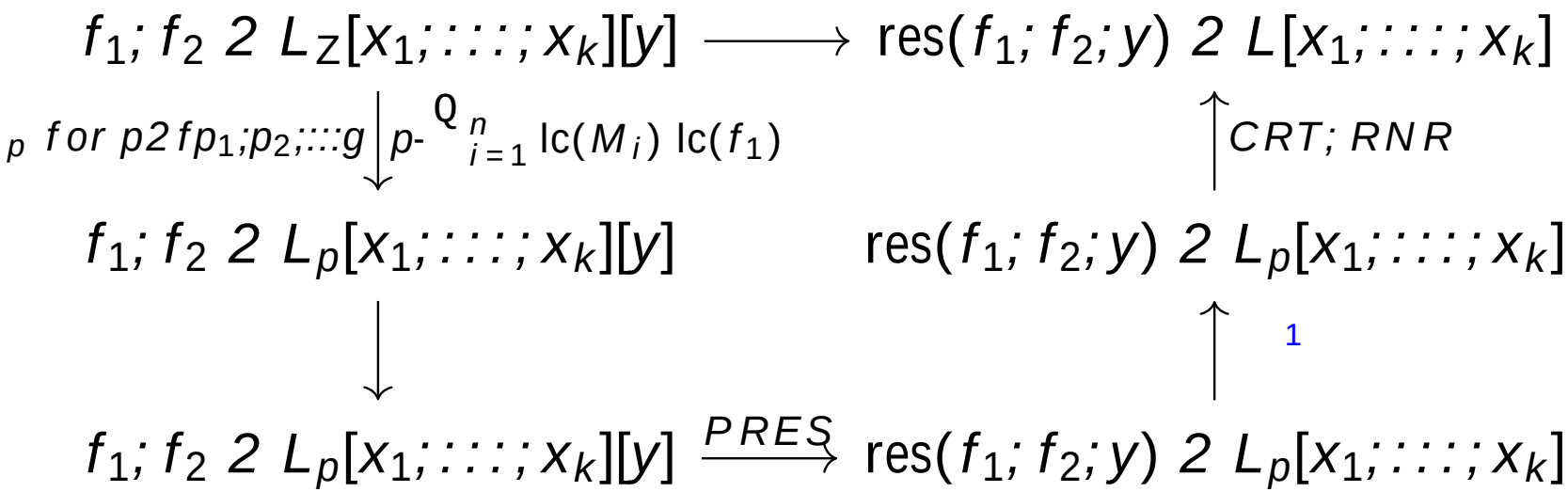
$$f_1 = 23z_2x + z_1y$$
$$f_2 = (z_2 + 5)x + z_1y$$

be two polynomials listed in the lexicographic order with $x > y$ over $L[x; y]$.

- $p_1 = 23 = ) \quad {}_{23}(\mathrm{lc}(f_1)) = 0 \mod p_1$. Thus $p_1$ is an lc-bad prime.
- $p_2 = 11 = ) \ \mathrm{lc}(f_2) = z_2 + 5$ and $z_2^2 - 3 \mod 11 = (z_2 + 5)(z_2 + 6)$ so $z_2 + 5$ is not invertible over $\mathbb{Z}_{11}[z_1; z_2]/\langle z_1^2 - 2; z_2^2 - 3i \rangle$.

## The isomorphism

: $L_p[x_1;:::;x_k]$ ! $L_p[x_1;:::;x_k]$ maps $f$ over $Z_p[z_1;:::;z_n]=\langle m_1(z_1);:::;m_n(z_n)\rangle$ to its corresponding polynomial over $Z_p[z]=\langle M(z)\rangle$.



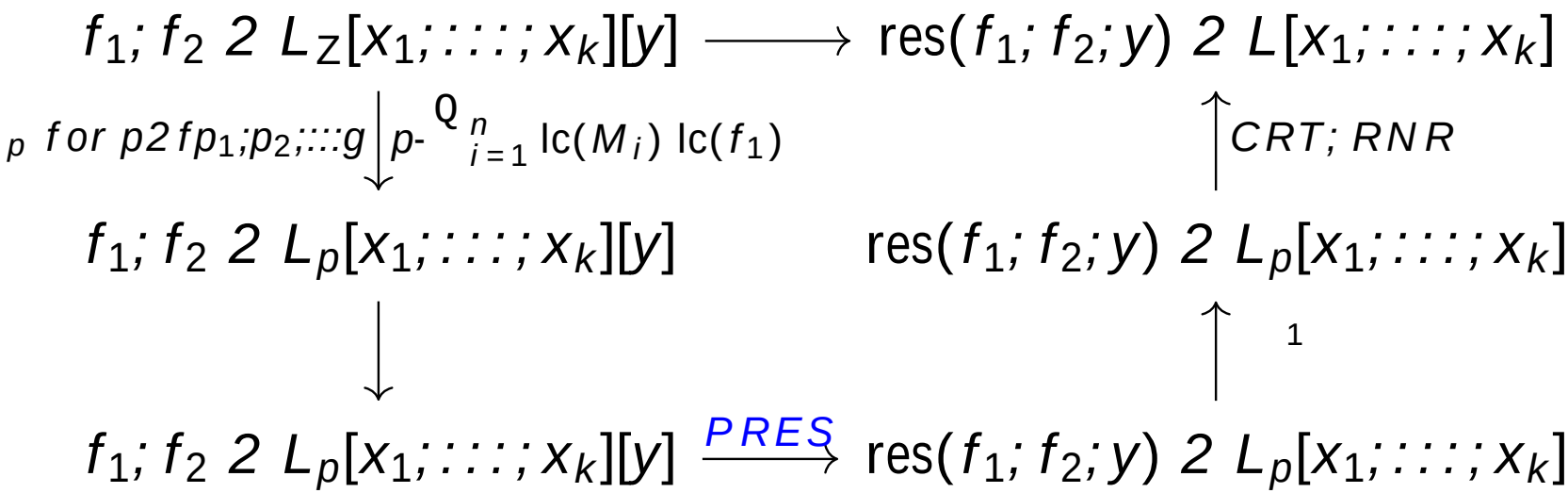

$L_p = Z_p[z_1;:::;z_n]=\langle m_1(z_1);:::;m_n(z_n)\rangle$ s.t $m_i(z_i) = M_i(z_i) \mod p$
$L_p = Z_p[z]=\langle M(z)\rangle$

For more detailed information on how works, please refer to our previous paper

[1] Ansari, Monagan
Computing GCDs of multivariate polynomi- als over algebraic number fields presented with multiple extensions.
*Computer Algebra in Scientific Computing, volume 14139 of LNCS, page 1–20. Springer, 2023.* .

# PRES

Algorithm PRES is a recursive algorithm to compute the $\mathsf{res}(f_1; f_2; y)$ where $f_1; f_2$ belong to $L_p[x_1; :::; x_k][y]$ and p is a prime.




If $k = 0$, then PRES calls the algorithm URES to compute $\mathsf{res}(f_1; f_2; y) \; 2 \; L_p$. PRES might fail.

PRES uses dense evaluation and interpolation to recover $x_1; :::; x_k$.

Question: Can we use any evaluation point?

Answer: No!!!

# Benchmark 1

Let $L = \mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{5}, \sqrt{7})$ have degree 16. the input polynomials $f_1$ and $f_2$ have degree $m$ in $x$ and $y$ and $\mathrm{res}(f_1, f_2, x)$ has degree $r_y$ in $y$: The coefficients of the input polynomials, $f_1$ and $f_2$, are polynomials in $z_1$, $z_2$, $z_3$, and $z_4$ with coefficients chosen randomly from $[1, 9)$:

| $m$ | $r_y$ | $N$ | MRES 1 | | | MRES 2 | |
|---|---|---|---|---|---|---|---|
| | | | time | | PRES | time | PRES |
| 2 | 4 | 4 | 0.313 | 0.126 | 0.140 | 0.828 | 0.828 |
| 4 | 16 | 4 | 0.828 | 0.187 | 0.501 | 8.609 | 8.563 |
| 6 | 36 | 7 | 3.938 | 0.189 | 3.218 | 59.938 | 59.610 |
| 8 | 64 | 11 | 14.171 | 0.218 | 11.891 | 291.281 | 289.875 |
| 10 | 100 | 16 | 47.500 | 0.468 | 48.842 | 967.437 | 962.609 |
| 12 | 144 | 22 | 119.766 | 0.596 | 103.016 | > 1000 | > 1000 |
| 14 | 196 | 29 | 282.844 | 0.798 | 244.189 | > 1000 | > 1000 |

# Benchmark 2

Let $f_1$ and $f_2$ be two polynomials in $L[x; y]$ where $L = \mathbb{Q}(\ _1; \ _2; \ _3)$. Let $M_1 = z_1^2 \quad 2$, $M_2 = z_2^2 \quad 3$, and $M_3 = \sum_{j=0}^{d_3} z_3^j + z_1 z_2$ be the minimal polynomials of $\ _1$, $\ _2$, and $\ _3$, respectively. Thus, $L$ is an algebraic number field of degree $d = 2 \quad 2 \quad d_3$. To consider various degrees for $L$, we change $d_3$.

| $d$ | $N$ | MRES 1 | | | MRES 2 | |
|---|---|---|---|---|---|---|
| | | time | | PRES | time | PRES |
| 16 | 5 | 43.688 | 0.095 | 42.562 | 288.265 | 287.811 |
| 24 | 5 | 55.203 | 0.156 | 53.688 | 379.735 | 379.077 |
| 32 | 5 | 57.234 | 0.249 | 55.517 | 513.797 | 513.078 |
| 40 | 5 | 67.719 | 0.375 | 65.641 | 628.547 | 627.361 |
| 48 | 5 | 80.687 | 0.624 | 78.094 | 745.578 | 744.703 |
| 56 | 5 | 100.953 | 0.922 | 97.031 | 894.734 | 893.921 |
| 64 | 5 | 114.062 | 1.375 | 110.171 | > 1000 | > 1000 |

# Complexity

Let $f_1; f_2 \; 2 \; L[x_1; \quad ; x_k; y]$.

Let $d$ be the degree of the number field $L$.

$T_f = \#f_1 + \#f_2$

$m = \deg(f_1; y)$, $n = \deg(f_2; y)$.

$D = (m + n)dx$ where $dx = \max_{i;j} \deg(f_i; x_j)$

We assume that multiplication and inverses in $L_p$ cost $O(d^2)$.

## Theorem

*Algorithm PRES does*

$$O(dD^k(T_f + mnd + kD)$$

*arithmetic operations in* $Z_p$.

Let $r = \mathrm{res}(f_1; f_2; y)$ and

$T_r = \#r$

$N$ is the number of good primes needed to reconstruct the resultant $r$.

$M = \log \max_{i=1}^{n} H(m_i)$.

$C = \log \max(H(f_1); H(f_2))$.

## Theorem

*Algorithm MRES does*

$$O(Nd(M + CT_M + d^2 + d(T_f + T_r) + D^k(T_f + mnd + kD) + NT_r))$$

*arithmetic operations.*

# Failure probability of lc-bad primes

Let $\mathsf{P}_{31} = \{\text{all 31 bit primes}\}$.
We have $\#\mathsf{P}_{31} = 50{,}697{,}537$.

## Theorem

*Let $f_1, f_2 \in L[x_1, \ldots, x_k, y]$ and*

1. $H = \max(\|\mathrm{lc}(f_1)\|_1, \|\mathrm{lc}(f_2)\|_1) < 2^h$,
2. $\mathrm{lc}(M_i) < 2^m$ *for* $1 \le i \le n$.

*If $p$ is chosen at random from $\mathsf{P}_{31}$ then*

$$\text{Prob}[p \text{ is an lc-bad prime}] \le \frac{2\lfloor \frac{h}{30} \rfloor + n\lfloor \frac{m}{30} \rfloor}{\#\mathsf{P}_{31}}.$$

# Failure probability of lc-bad evaluation point

**Theorem**

*Let* $2\ Z_p^k$ *be chosen at random, then*

$$\text{Prob}[\ \textit{is an lc-bad evaluation point}] \quad \frac{\deg(f_1)}{p}.$$

# Failure probability of Det-bad Primes

In our paper we considered the case where $m_i \in \mathbb{Z}[z_1, \ldots, z_i]$ are monic for $1 \leq i \leq n$ so $A \in \mathbb{Z}^{d \times d}$.

## Theorem

*Let* $= z_1 + C_1 z_2 + \cdots + C_{n-1} z_n$ *where* $0 \leq C_i \in \mathbb{Z}$ *for* $1 \leq i \leq n-1$ *and* $k \in k_1 \subset \mathbb{C}$. *Suppose* $\det(A) \neq 0$. *If* $p$ *is chosen at random from* $P_{31}$ *then*

$$\text{Prob}[p \mid \det(A)] \leq \frac{b(d/2 \log_2 d + d(C + \sum_{i=1}^{n} \alpha_i \log_2(1 + D_i k m_{n-i+1} k_1)))c}{30 \cdot \#P_{31}}.$$

*where* $D_i = \frac{d}{\prod_{j=1}^{i} d_{n-j+1}}$, $\alpha_1 = d - d_n + 1$, *and*

$\alpha_i = d - d_{n-i+1} + 1 + (d_{n-i+1} - 1) \sum_{j=1}^{i-1} \alpha_j$ *for* $2 \leq i \leq n$.

Figura: Portrait of Dr. Gachet, Van Gogh, 1890, France